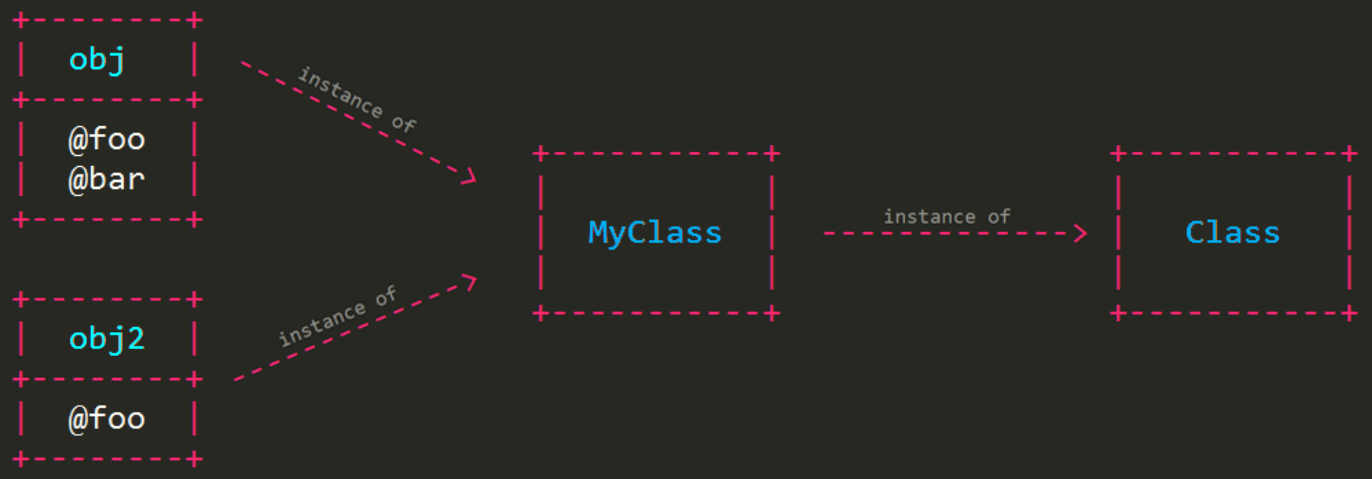
Insight:

Where variables & methods live?

Unlike other programming languages, ruby classes don't have class properties. Instance variables just spring into existence when they are assigned a value. Class instance variables reside in objects, methods in the class of the object.

Where instance variables reside?

```ruby
class MyClass
    attr_reader :foo, :bar

    def initialize
        @foo = "foo"

    def big_bang
        @bar = "bar"
    end
end

obj  = MyClass.new
obj2 = MyClass.new

obj.foo                         # "foo"
obj.bar                         # nil
obj.instance_variables          # [:@foo]

obj.big_bang

obj.bar                         # "bar"
obj.instance_variables          # [:@foo, :@bar]
obj2.instance_variables         # [:@foo]
```



Where class instance variables reside?

```ruby
class MyClass
    @foo = "class instance variable"

    def initialize
        @bar = "instance variable"
    end
end

obj = MyClass.new
```

Quick recap:

- Instance variables in classes are only created when they are explicitly assigned to a value
- Instance variables live in the corresponding objects, not classes
- Ruby classes do not have predefined class properties. Each instance of a class may have its' own set of variables.

Where methods reside?

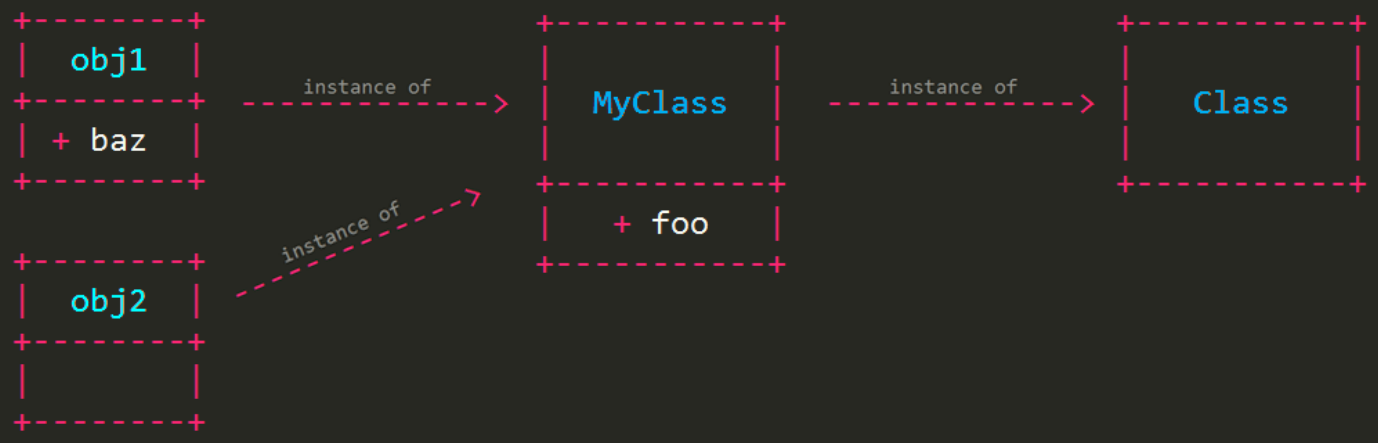
```ruby
class MyClass
    def foo
        "foo"
    end
end

obj = MyClass.new

obj.methods - Object.new.methods          # [:foo]
MyClass.instance_methods(false)           # [:foo]


class MyClass
    def bar
        "bar"
    end
end

obj.methods - Object.new.methods          # [:foo, :bar]
```

Where class/eigenclass/static methods live?

```ruby
class MyClass
    def foo
    end

    def self.bar
    end

    class << self
        def baz
        end
    end
end

obj = MyClass.new
```

Where singleton methods live?

```ruby
class MyClass
    def foo
        "foo"
    end
end

obj1 = MyClass.new
obj2 = MyClass.new

def obj1.baz
    "baz"
end

obj1.foo                                    # "foo"
obj2.foo                                    # "foo"

obj1.baz                                    # "baz"
obj2.baz                                    # NoMethodError: undefined method `baz' for #<MyClass:0x007f9

methods_inherited_from_object = Object.instance_methods

obj2.methods - methods_inherited_from_object    # []
obj1.methods - methods_inherited_from_object    # [:baz]
```



Quick recap:

- Methods live in classes
- Static/class/eigenclass methods live in eigenclass of the class
- Singleton methods reside in objects

Insight:

Where variables & methods live?

Instance variables just spring into existence when they are assigned a value.
Class instance variables reside in objects and methods in the class of the object.